LA-UR 95-3348

Approved for public release; distribution is unlimited

Complexity of Hierarchically and One-dimensional Periodically-specified Problems I: Hardness Results

Authors: M.V. Marathe, H.B. Hunt III, R.E. Stearns
V. Radhakrishnan

DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume **00**, 19xx

Complexity of Hierarchically and 1-Dimensional Periodically Specified Problems I: Hardness Results

Madhav V. Marathe, Harry B. Hunt III, Richard E. Stearns, and Venkatesh Radhakrishnan

Abstract. We study the complexity of various combinatorial problems when instances are specified using one of the following succinct specifications: (1) the 1-dimensional finite periodic narrow specifications (denoted 1-FPN-specifications) of Ford et al. and Wanke [**FF58, Wa93**]; (2) the 1-dimensional finite periodic narrow specifications with explicit boundary conditions (denoted 1-FPN(BC)-specifications) of Gale [**Ga59**]; (3) the 2-way infinite 1-dimensional periodic narrow specifications (denoted 1-PN-specifications) of Orlin et al. [**Or82a**]; and (4) the hierarchical specifications (denoted L-specifications) of Lengauer et al. [**LW87a**].

We present three general types of results. First, we give a polynomial time algorithm that, given a 1-FPN- or 1-FPN(BC)-specification of a graph (or a CNF formula), constructs a level-restricted L-specification of an isomorphic graph (or formula). Second, we prove the PSPACE-hardness of several basic CNF satisfiability problems when instances are specified succinctly using 1-FPN-, 1-FPN(BC)-, 1-PN- or L-specifications. Finally, we use our first two types of results to prove the PSPACE-hardness of a number of basic combinatorial problems when instances are so specified. These results along with those in [**MH+96**] include all the problems shown to be PSPACE-hard for 1-PN-specifications by Orlin [**Or82a**] and for L-specifications by Lengauer and Wagner [**LW92**]. Our results significantly extend the hardness results in [**Or82a, Pa94, LW92, Wa93**].

1991 *Mathematics Subject Classification.* 68Q25, 68Q15, 68R10.

Key words and phrases. Hierarchical and Periodic Specifications, Computational Complexity, PSPACE-hardness, Satisfiability Problems.

The work is supported by the Department of Energy under Contract W-7405-ENG-36.

This research was supported in part by NSF Grants CCR-94-06611 and CCR-90-06396.

This research was supported in part by NSF Grants CCR-94-06611 and CCR-90-06396.

1. Introduction

Many practical applications of graph theory and combinatorial optimization in CAD systems, mechanical engineering, VLSI design, transportation networks and software engineering involve processing large (but regular) objects constructed in a systematic manner from smaller and more manageable components. Consequently, the graphs that abstract the structure and operation of the underlying circuits (designs) also have a regular structure and are defined in a systematic manner using smaller graphs. Such methods for specifying large and regular objects by small specifications are referred to as *succinct specifications*. Here we consider two of these specifications, namely (i) hierarchical specifications [**Ga82, GW83, LW92, BOW83, RH93**], and (ii) periodic specifications [**CM91, HW95, IS87, KO91, KS88, Or82a, Wa93**].

Hierarchical specifications are useful in describing large scale systems with very regular structures. This is because such specifications allow the overall design of an object to be partitioned into the design of a collection of modules which is a much more manageable task than producing a complete design in one step. We refer the reader to [**Le86, Ga82, RH93, Ma94, HLW92**] for a detailed discussion on this topic. Consequently hierarchical specifications are widely used in areas such as VLSI design and analysis, finite element analysis, software engineering and datalog queries (see [**HLW92, Ma94**] and the references therein). Due to their wide spread applicability, researchers have extensively studied hierarchically specified problems [**BOW83, Wa86, Ga82, LW87a, LW92, LW93**]. In this paper we study the hierarchical specifications of Lengauer et al. [**LW87a, LW92**].

Periodic specifications can also be used to define large scale systems with highly regular structures. Using periodic specifications, large objects are described as repetitive connections of a basic module. Frequently, the modules are connected in a straight line; however the basic modules can also be repeated in two or higher dimensional patterns. Two dimensional periodic specifications arise naturally in the study of regular VLSI circuits such as systolic arrays and VLSI signal processing arrays [**CS81, IS86**]. One dimensional periodic specifications are used to model time variant problems, where the constraints or demands for any one period is the same as those for preceding or succeeding periods. These specifications have applications in such diverse areas as transportation planning [**Or82a, HLW92, Ma94**], parallel programming [**HLW92, KMW67**], real time computer aided verification and VLSI design [**IS87, IS88**]. Here we study the one dimensional periodic specifications of infinite objects of Orlin [**Or82a**] and the periodic specifications of finite objects of Ford and Fulkerson [**FF58**], Wanke [**Wa93**] and Gale [**Ga59**].

Generally speaking both hierarchical and periodic specifications are extensions of the standard specifications used to represent instances. An important feature of both of these kinds of specifications is that they can be much more concise in describing objects than standard specifications. In particular, periodic or hierarchical specif¡ications of size $\Theta(n)$ can represent objects of size $2^{\Omega(n)}$. The complexity of solving a problem is usually measured as a function of the size of the specification of the problem instance. Therefore, the complexity of a problem when instances are specified using standard descriptions can be different when compared to the complexity of the same problem when instances are specified hierarchically or periodically. For example, while the 3-COLORING problem is NP-complete when the graphs are represented by adjacency matrices or by adjacency lists [**GJ79**], it is PSPACE-complete when instances are specified by the hierarchical specifications of Lengauer [**LW92**] or by 1-dimensional periodic specifications of Orlin [**Or82a**]. In contrast, the 2-COLORING problem is solvable in polynomial time *even* when instances are specified by the hierarchical specifications of Lengauer et al. [**LW92, LW87a**] or by the 1-dimensional periodic specifications of Orlin [**Or82a**].

In the past, satisfiability problems have been used to model a number of problems in areas such as automated reasoning, computer-aided design [**GP+96b**], computer-aided manufacturing [**GP+96a**], machine vision [**Gu92**], database, robotics, integrated circuit design [**GP+96a, GP+96b**], computer architecture design and computer network design, etc. We refer to [**GP+96a**] for an excellent recent survey on this topic. Similarly, as has been amply demonstrated in the past, satisfiability problems serve as rich collection of basic problems used to prove NP-hardness of other combinatorial problems (see [**GJ79, Sc78**]). Here we show that analogously, succinctly specified satisfiability problems also serve two important purposes: (i) they are useful in modeling problems arising in practical applications and (ii) serve as tools for obtaining unified easiness/hardness results for succinctly specified problems.

2. Summary of Results

Here, we study the complexity of a number of combinatorial, graph and generalized CNF satisfiability problems, when instances are specified using one of the following specifications:

1. the 1-dimensional finite periodic narrow specifications (denoted 1-FPN-specifications) of Ford and Fulkerson [**FF58**] and of Wanke [**Wa93**];
2. the 1-dimensional finite periodic narrow specifications with explicit boundary conditions (denoted 1-FPN(BC)-specifications) of Gale [**Ga59**] and others;
3. the 2-way infinite 1-dimensional narrow periodic (sometimes called dynamic) specifications (denoted 1-PN-specifications) of Karp et al. and Orlin [**KMW67, Or82a**]; and
4. the hierarchical specifications (denoted L-specifications) of Lengauer [**LW87a, LW92**].

Let Π be a problem, whose instances are specified *non-succinctly* using one of the standard specifications in the literature. For example, instances of CNF satisfiability problems are specified non-succinctly by CNF formulas and by sets of clauses, each clause being a set of literals. Here, we use 1-FPN-, 1-FPN(BC)-, 1-PN- and L-Π to denote the problem Π when its instances are specified succinctly by 1-FPN-, 1-FPN(BC)-, 1-PN- and L- specifications, respectively. We use α-Π to denote the problems 1-FPN-, 1-FPN(BC)-, 1-PN- and L-Π; and we use *succinct* specification to mean 1-FPN-, 1-FPN(BC)-, 1-PN- and L-specification. Thus for example, 1-FPN-3SAT denotes the problem 3SAT when instances are specified by 1-FPN-specifications; and α-3SAT denotes the problems 1-FPN-, 1-FPN(BC)-, 1-PN- and L-3SAT. Our results are summarized below.

2.1. Complexity of Satisfiability Problems.

In Theorem 4.1(called the **Translation Theorem**), we prove that a 1-FPN- or 1-FPN(BC)- specification of a graph or formula can be translated in polynomial time into an L-specification of an isomorphic graph or formula, respectively. This implies that, for many problems Π including all problems considered here, the problems 1-FPN- or 1-FPN(BC)-Π are polynomial time reducible to the problem L-Π.

In Section 5, we prove the PSPACE-hardness of several CNF satisfiability problems when inputs are specified succinctly. In particular, we prove that the problem 1-FPN-3SAT is PSPACE-hard. By the Translation Theorem, this implies that L-3SAT is also PSPACE-hard.

2.2. Applications. In Section 6, we outline how a polynomial time reduction involving local replacement [**GJ79**] from the problem 3SAT, NAE-3SAT, etc., to a problem Π can be extended to obtain a polynomial time reduction from the corresponding problem α-3SAT, α-NAE-3SAT , etc., to the problems α-Π. Together with the results outlined in Sections 2.1 and 2.2, this enables us to do the following:

1. derive alternative and unified proofs of the PSPACE-hardness results for 1-PN- specifications in [**Or82a**] and for L-specifications in [**LW92**]; and
2. derive a number of new PSPACE-hardness results for succinctly specified problems.

Often, our PSPACE-hardness proofs involve reductions that are more time and/or space efficient than those in [**Or82a, LW92**]. Most of these hardness results hold, even when restricted to specifications of $O(\log \mathcal{N})$ bandwidth-bounded graphs. For the rest of the paper we use $\mathcal{N}$ to denote the size of expanded objects (graphs or formulas).

2.3. Comparisons with Related Work. Orlin [**Or82a**] proved that the problem 1-PN-3SAT is PSPACE-complete. He used this and known reductions from 3SAT to prove the PSPACE-hardness of the problems

> 1-PN-KNAPSACK, 1-PN-HAMILTONIAN-PATH(1-PN-HP),
> 1-PN-3-COLORING, and 1-PN-3DM.

Wanke [**Wa93**] has also proved PSPACE-hardness results for periodically-specified problems; but his results do not hold for 1-dimensional periodically-specified problems. Lengauer and Wagner [**LW92**] have proved the PSPACE-hardness of the problems

> L-3-COLORING, L-INDEPENDENT SET(L-IS),
> L-HAMILTONIAN CIRCUIT(L-HC), L-MONOTONE CIRCUIT
> VALUE PROBLEM(L-MCVP), L-NETWORK FLOW(L-NF), and
> L-ALTERNATING GRAPH ACCESSIBILITY PROBLEM
> (LAGAP).

Their PSPACE-hardness results for the problems L-3-COLORING, L-IS, and L-HC hold for $O(\log \mathcal{N})$ bandwidth-bounded instances. Several PSPACE-hardness results for L-specified unit disk graphs were presented in our paper [**MR+93**].

Our results presented here extend the above results proved in [**Or82a, LW92**] in the following several ways:

1. Previously, the complexities of 1-PN- and L-specified problems have been studied separately. We show that there is a close correspondence between Orlin's PSPACE-hardness results for 1-PN-specified problems and Lengauer and Wagner's PSPACE-hardness results for L-specified problems.
2. The only previous work on the complexities of the problems α-SAT(S) is that of Orlin [**Or82a**] on the PSPACE-hardness of the problem 1-PN-3SAT. (Several references to the results in this paper occur in our papers [**MR+93, MH+93a, MH+94**].)
3. No PSPACE-hardness results for succinctly presented planar problems have been presented previously. No PSPACE-hardness results of any kind have been presented previously, for problems specified succinctly by either 1-FPN- or 1-FPN(BC)-specifications.
4. Several of our reductions used to prove PSPACE-hardness are more efficient in both time and space than the corresponding reductions in [**Or82a, LW92**]. For example, our proof of the PSPACE-hardness of 1-PN-3SAT is by an $O(n \log n)$ time reduction of the acceptance problem for an arbitrary nondeterministic LBA. That of Orlin [**Or82a**] is by an $\Omega(n^2 \log n)$ time and space reduction of the same problem. Under the plausible assumption that there exist LBAs whose acceptance problems require $2^{\Omega(n)}$ time on deterministic Turing machines, our reduction implies that the problem 1-PN-3SAT requires $2^{\Omega(\frac{n}{\log n})}$ time on deterministic Turing machines. That of Orlin only implies that the problem 1-PN-3SAT requires $2^{\Omega(\frac{\sqrt{n}}{\log n})}$ time on deterministic Turing machines.

The rest of the paper is organized as follows. Section 3 gives both the definitions and examples of the kinds of succinct specifications considered. In Section 4 we prove a theorem relating the 1-FPN- and 1-FPN(BC)-specifications to equivalent L-specifications. In Section 5, we prove the PSPACE-hardness of the problems 1-FPN-3SAT, 1-PN-3SAT and L-3SAT. In Section 6, we illustrate the applications of our results in Section 5, to prove PSPACE-hardness of a number of combinatorial problems specified succinctly. These problems include the various PSPACE-hard problems for 1-PN-specifications in [**Or82a**] and for L-specifications in [**LW92**].

3. Definitions and Preliminaries

Sections 3.1 through 3.5 give basic definitions used here, including the definitions of the generalized CNF satisfiability problems of [**Sc78**] and the kinds of succinct specifications considered. We illustrate these definitions with several examples. Other basic definitions and concepts can be found in [**GJ79, CLR, AHU**].

3.1. The Problems SAT(S).

We first review some basic definition from Schaefer [**Sc78**].

DEFINITION 3.1. *(Schaefer* [**Sc78**]*)*
Let $\mathsf{S} = \{R_1, R_2, \cdots, R_m\}$ *be a finite set of finite arity boolean relations. (A boolean relation is defined to be any subset of* $\{0,1\}^p$ *for some integer* $p \geq 1$. *The integer* p *is called the* **rank** *of the relation.) An* S*-formula is a conjunction of clauses each of the form* $\hat{R}_i(\xi_1, \xi_2, \cdots)$, *where* $\xi_1, \xi_2, \cdots$ *are distinct, unnegated variables whose number matches the rank of* $R_i, i \in \{1, \cdots m\}$ *and* $\hat{R}_i$ *is the relation symbol representing the relation* R_i. *The* S*-satisfiability problem (denoted by* SAT(S)*) is the problem of deciding whether a given* S*-formula is satisfiable.*

The problem $\mathsf{SAT}_c(\mathsf{S})$ *is the variant of the problem* SAT(S) *in which the constants* 0 *and* 1 *are also allowed to occur in* S*-formulas.*

The generalized CNF satisfiability problems SAT(S) and $\mathsf{SAT}_c(\mathsf{S})$ generalize the problems 3SAT, 1-3SAT , NAE-3SAT, etc. [**GJ79**]. For example, let $R(x, y, z)$ be the ternary logical relation given by $\{(1,0,0),(0,1,0),(0,0,1)\}$. Then, the problem 1-3SAT is the same as the problem SAT({R}). Let f be an S-formula with m clauses and n_i literals in clause i, $1 \leq i \leq m$. The size of f denoted by $size(f)$ is given by $O(\sum_{i=1}^{i=m} n_i)$.

DEFINITION 3.2. *The logical relation* R *is* **weakly positive** *if* $R(x_1, x_2, \ldots)$ *is logically equivalent to some CNF formula having at most one negated variable in each conjunct.*
The logical relation R *is* **weakly negative** *if* $R(x_1, x_2, \ldots)$ *is logically equivalent to some CNF formula having at most one unnegated variable in each conjunct.*

3.2. Definitions from Graph Theory.

Let $G(V, E)$ be a finite undirected graph. A one-to-one function $f : V \rightarrow \{1, \ldots, |V|\}$ is called a *layout* of G. Following [**MS81**] we say that G has *bandwidth* k under the layout f, if, for all edges $(x,y) \in E$, $|f(x) - f(y)| \leq k$. The concept of bandwidth also applies to CNF formulas as follows. Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a CNF formula. We say that F has bandwidth k if there exists a permutation σ of $\{1, 2, \ldots, m\}$ such that if two clauses C_i and C_j contain the same variable (negated or unnegated) then $|\sigma(i) - \sigma(j)| \leq k$. Recall that a graph is *planar* if it can be laid out in the plane without crossover of its edges.

Definition 3.3. *Let F be an instance of the problem 3SAT with set of variables V and set of clauses C.*

1. The bipartite graph of f, denoted $\mathbf{BG}(f)$, is the bipartite graph $(V \cup C, E)$, where $e = (c,v) \in E$ if and only if variable v occurs in clause c.

2. f is said to be planar if and only if the graph $\mathbf{BG}(f)$ is planar.

3.3. L-specified Graphs. The definition of an L-specification , of a graph, of the graph $E(,)$ specified by , , and of the hierarchy tree $HT(,)$ of , closely follow [**Le89**].

Definition 3.4. *An L-specification , $= (G_1, \ldots, G_n)$ of a graph is a sequence of labeled undirected simple graphs G_i called cells. The graph G_i has m_i edges and n_i vertices. p_i of the vertices are called pins. The other $(n_i - p_i)$ vertices are called inner vertices. r_i of the inner vertices are called nonterminals. The $(n_i - r_i)$ vertices are called terminals. The remaining $n_i - p_i - r_i$ vertices of G_i that are neither pins nor nonterminals are called explicit vertices.*

Each pin of G_i has a unique label, its name. The pins are assumed to be numbered from 1 to p_i. Each nonterminal in G_i has two labels (v,t), a name and a type. The type t of a nonterminal in G_i is a symbol from $G_1, \ldots, G_{i-1}$. All the neighbors of a nonterminal vertex must be terminals. If a nonterminal vertex v is of the type G_j in G_i, then letting v be of degree p_j, each terminal vertex that is a neighbor of v has a distinct label (v,l) such that $1 \leq l \leq p_j$. We say that the neighbor of v labeled (v,l) matches *the lth pin of G_j.*

Note that a terminal vertex may be a neighbor of several nonterminal vertices. Given an L-specification , , $N = \sum_{1 \leq i \leq n} n_i$ denotes the *vertex number*, and $M = \sum_{1 \leq i \leq n} m_i$ denotes the *edge number* of , . The size of , , denoted by $size(,)$, is $N + M$.

Definition 3.5. *Let , $= (G_1, \ldots, G_n)$ be an L-specification of a graph $E(,)$ and let ,$_i = (G_1, \ldots, G_i)$. The expanded graph $E(,)$ (i.e. the graph*

associated with ,) is obtained as follows:
$k = 1 : E(,) = G_1$.
$k > 1$: *Repeat the following step for each nonterminal v of G_k, say of the type G_j: delete v and the edges incident on v. Insert a copy of $E(, _j)$ by identifying the l^{th} pin of $E(, _j)$ with the node in G_k that is labeled (v, l). The inserted copy of $E(, _j)$ is called a subcell of G_k.*

Observe that the expanded graph can have multiple edges although none of the G_i have multiple edges. Here however, we **only** consider **simple** graphs, i.e. there is at most edge between a pair of vertices. This means that multi edges are treated simply as single edges. We assume that , is not redundant in the sense that for each i, $1 \leq i \leq n$, there is a nonterminal v of type G_i in the definition of G_j, $j > i$.

The expansion $E(,)$ is the graph associated with the L-specification , with vertex number N. For $1 \leq i \leq n$, $, _i = (G_1, ..., G_i)$ is the L-specification of the graph $E(, _i)$. Note that the total number of nodes in $E(,)$ can be $2^{\Omega(N)}$. (For example, a complete binary tree with $2^{\Omega(N)}$ nodes can be specified using an L-specification of size $O(N)$.) To each L-specification $, = (G_1, ..., G_n)$, $(n \geq 1)$, we associate a labeled rooted unoriented tree $HT(,)$ depicting the insertions of the copies of the graphs $E(, _j)$ $(1 \leq j \leq n-1)$, made during the construction of $E(,)$ as follows:

Definition 3.6. *Let $, = (G_1, ..., G_n)$, $(n \geq 1)$ be an L-specification of a graph. The* **hierarchy tree** *of , , denoted by $HT(,)$, is the labeled rooted unordered tree defined as follows:*

1. *Let r be the root of $HT(,)$. The label of r is G_n. The children of r in $HT(,)$ are in* one-to-one *correspondence with the nonterminal vertices of G_n as follows: The label of the child s of r in $HT(,)$ corresponding to the nonterminal vertex (v, G_j) of G_n is (v, G_j).*
2. *For all other vertices s of $HT(,)$ and letting the label of $s = (v, G_j)$, the children of s in $HT(,)$ are in* one-to-one *correspondence with the nonterminal vertices of G_j as follows: The label of the child t of s in $HT(,)$ corresponding to the nonterminal vertex (w, G_l) of G_j is (w, G_l).*

Given the above definition, we can naturally associate a hierarchy tree with each $, _i$, $1 \leq i \leq n$. We denote this tree by $HT(, _i)$. Note that, each vertex v of $E(,)$ is either *an explicit vertex of G_n* or *is the copy of some explicit vertex v' of G_j $(1 \leq j \leq n)$ in exactly one copy C_j^w of the graph*

$E(, _j)$ inserted during the construction of $E(,)$. This enables us to assign v of $E(,)$ to the unique vertex n_w of the $HT(,)$ given by

1. if v is a terminal vertex of G_n, then n_w is the root of $HT(,)$, and
2. otherwise, v belongs to the node n_w that is the root of the hierarchy tree $HT(, _j)$, corresponding to C_j^w.

Given $HT(,)$, the *level number* of a node in $HT(,)$ is defined as the length of the path from the node to the root of the tree.

As noted in [**Le89**], L-specifications have the property that for each copy (instance) of a subcell, a complete boundary description has to be given. Thus if a nonterminal has many pins, copying it is costly. Another property of the definition of L-specifications is that nonterminals are adjacent only to terminals. These properties ensure that the size of the "frontier" (or the number of neighbors) of any nonterminal is polynomial in the size of the specification. As a result, regular structures such as grids cannot be specified using small (logarithmic sized in terms of the object) L-specifications. (see [**LW87a**]). In contrast, in the graph glueing model of Galperin [**Ga82**] the size of the frontier can be exponentially large. Consequently using this model, graphs such as grids can be represented using descriptions of logarithmic size.

As demonstrated in [**Le89, LW87a, Le88, Ga82, Wa86**], the explicit description of boundaries and non-existence of edges between non-terminals allow the construction of efficient algorithms for L-specified problems. The size of the frontier also has a significant impact on the complexity of several basic succinctly specified problems. For example, several basic NP-hard problems become PSPACE-hard when specified using L-specifications (see [**LW92**] and the results in this paper). In contrast, in a recent paper we show that these problems typically become NEXPTIME-hard when specified using the graph glueing specifications of [**Ga82**] (see [**MH+95c**]).

By Definition 3.4, it follows that an L-specification is a restricted form of context-free graph grammar. The substitution mechanism glues the pins of cells to neighbors of nonterminals representing these cells, as described in Definition 3.5. Such graph grammars are known as *hyperedge replacement systems* [**HK87**] or *cellular graph grammars* [**LW93**]. Two additional restriction are imposed on cellular graph grammars to obtain L-specified graphs. First, for each nonterminal there is only one cell that can be substituted. Thus there are no *alternatives* for substitution. Second, the index of the substituted cell has to be *smaller* than the index of the cell in which the

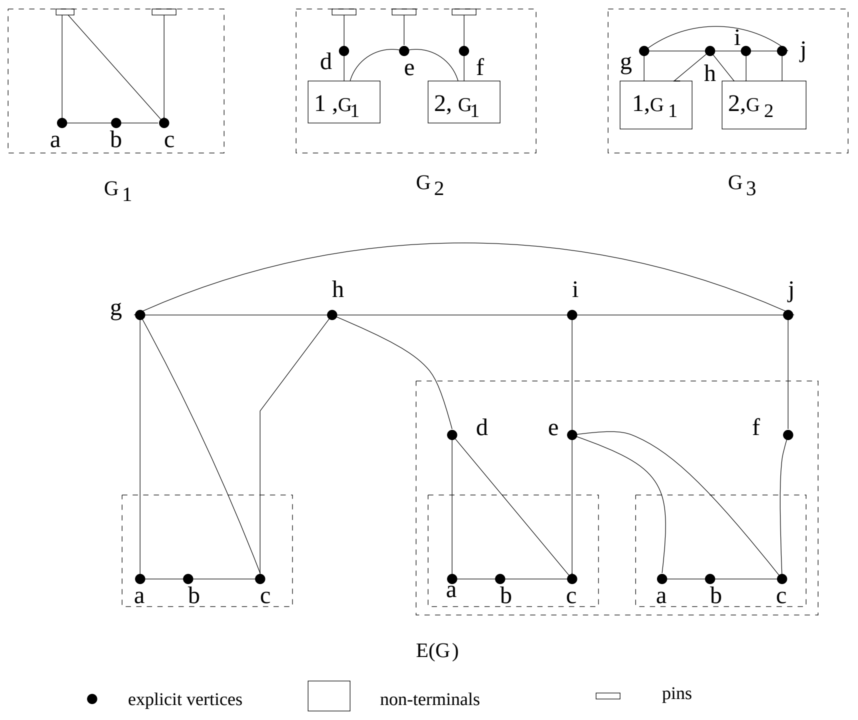

FIGURE 1. An L-specification $G = (G_1, G_2, G_3)$ of a graph $E(G)$.

nonterminal occurs. The acyclicity condition together with the condition that there are no alternatives implies that an L-specification defines a finite and unique graph. We observe that $HT(,)$ is the parse tree of the unique graph generated by the context-free graph grammar , .

EXAMPLE 3.7. *Let $G = (G_1, G_2, G_3)$ be the L-specification of a graph given in Figure 1. The the graph $E(G)$ is shown in Figure 1. The one-to-one correspondence between the pins and their neighbors is clear from the figure and hence is not given explicitly. The hierarchy tree $HT(G)$, associated with G is shown in Figure 2.*

3.4. Level restricted specifications. Next, we discuss level restricted L-specifications also defined in [**MR+93, MH+93a**].

DEFINITION 3.8. *An L-specification , $= (G_1, ..., G_n)$ $(n \geq 1)$, of a graph G is 1-level-restricted, if $\forall$ $(u,v) \in E(,)$, $u \in n_u$ and $v \in n_v$ of $HT(,)$, either*
(1) n_u and n_v are the same vertex of $HT(,)$, or
(2) one of n_u or n_v is the parent of the other in $HT(,)$.

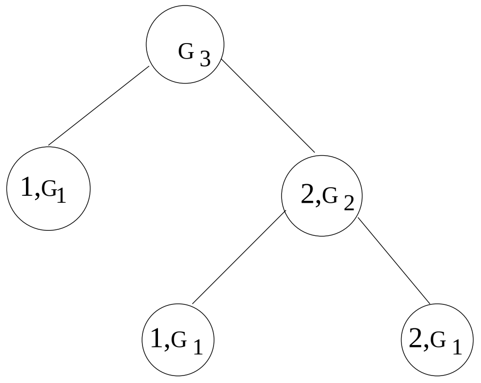

FIGURE 2. The hierarchy tree $HT(G)$ associated with G.

Extending the above definition we can define k-level-restricted specifications. An L-specification , $= (G_1, ..., G_n)$, $(n \geq 1)$, of a graph $E(,)$ is *k-level-restricted*, if for all edges (u, v) of $E(,)$, either
(1) n_u and n_v are the same vertex of $HT(,)$ or
(2) one of n_u or n_v is an ancestor of the other in $HT(,)$ and the length of the path between n_u and n_v in $HT(,)$ is no more than k.

We note that for any fixed $k \geq 1$, k-level-restricted L-specifications can still lead to graphs that are exponentially large in the sizes of their specifications. Moreover, L-specifications for several practical designs in [**Le82, Le86, LW87a**] are *k-level restricted* for small values of k. (For example, it is easy to specify a complete binary tree with $2^{\Omega(N)}$ nodes by a 1-level restricted L-specification of size $O(N)$.)

3.4.1. *L-specified S-formulas.* Let S be a finite nonempty set of finite-arity Boolean relations. We extend the definition of S-formula, SAT(S) and SAT_c(S) to specify L-specifications of an S-formula, the problems L-SAT(S) and L-SAT_c(S). These definitions closely follow Definitions 3.4, 3.5 and 3.1 respectively.

DEFINITION 3.9. *An instance $F = (F_1(X^1), \ldots, F_{n-1}(X^{n-1}), F_n(X^n))$ of* L-SAT(S) *is of the form*

$$F_i(X^i) = (\bigwedge_{1 \leq j \leq l_i} F_{i_j}(X^i_j, Z^i_j)) \bigwedge f_i(X^i, Z^i)$$

for $1 \leq i \leq n$ where f_i is an S-formula, $X^n = \phi$, X^i, X^i_j, Z^i, Z^i_j, $1 \leq i \leq n-1$, are vectors of Boolean variables such that $X^i_j \subseteq X^i$, $Z^i_j \subseteq Z^i$, $0 \leq i_j < i$. Thus, F_1 is just a S-formula formula. An instance of L-SAT(S) specifies a CNF formula $E(F)$, that is obtained by expanding the F_j,

$2 \leq j \leq n$, where the set of variables Z's introduced in any expansion are considered distinct. The problem L-SAT(S) *is to decide whether the formula $E(F)$ specified by F is satisfiable.*

Extending the definition of $\mathsf{SAT}_c(\mathsf{S})$ along the lines of Definition 3.9, we can define the problems L-$\mathsf{SAT}_c(\mathsf{S})$. Thus we omit the formal statement of these problems here. Let n_i be the total number of variables used in F_i (i.e. $|X^i| + |Z^i|$) and let m_i be the total number of clauses in F_i. The size of F, denoted by $size(F)$, is equal to $O(\sum_{1 \leq i \leq n} (m_i n_i))$. Given a formula $E(F)$ specified by an L-specification F, $BG(E(F))$ denotes the bipartite graph associated with $E(F)$. We use $H[BG(E(F))]$ to denote the L-specification of $BG(E(F))$.

It is easy to extend the concept of level restricted L-specifications for graphs to define level restricted L-specified formulas. Therefore, we omit the formal definition here.

EXAMPLE 3.10. *Let $F = (F_1(x_1, x_2), F_2(x_3, x_4), F_3)$ be an instance of* L-3SAT *where each F_i is defined as follows:*

$$F_1(x_1, x_2) = (x_1 \vee x_2 \vee z_1) \wedge (z_2 \vee z_3)$$

$$F_2(x_3, x_4) = F_1(x_3, z_4) \wedge F_1(z_4, z_5) \wedge (z_4 \vee z_5 \vee x_4)$$

$$F_3 = F_1(z_7, z_6) \wedge F_2(z_8, z_7)$$

By substituting F_1 in the definition of F_2 we get

$$E(F_2(x_3, x_4)) = (x_3 \vee z_4 \vee z_1^1) \wedge (z_2^1 \vee z_3^1) \wedge (z_4 \vee z_5 \vee z_1^2) \wedge (z_2^2 \vee z_3^2) \wedge (z_4 \vee z_5 \vee x_4)$$

Using this, it can be seen that the formula $E(F)$ denoted by F is $(z_7 \vee z_6 \vee z_1^1) \wedge (z_2^1 \vee z_3^1) \wedge$ $(z_8 \vee z_4 \vee z_1^2) \wedge (z_2^2 \vee z_3^2) \wedge$ $(z_4 \vee z_5 \vee z_1^3) \wedge (z_2^3 \vee z_3^3) \wedge (z_4 \vee z_5 \vee z_7)$. $\square$

DEFINITION 3.11. *The problem* L-SAT(S) *(or* L-$\mathsf{SAT}_c(\mathsf{S})$*) is the problem of determining, given an* L*-specification F of the* S*-formula(or of the* S*-formula with constants), if $E(F)$ is satisfiable.*

DEFINITION 3.12. *The problem* L-Pl-3SAT *is to decide whether the planar* 3CNF *formula $E(F)$ specified by an* L*-specification F is satisfiable.*

The problem L-Pl-SAT(S) is defined analogously. In general, we use L-Pl-Π to denote the problem Π restricted to planar L-specified instances.

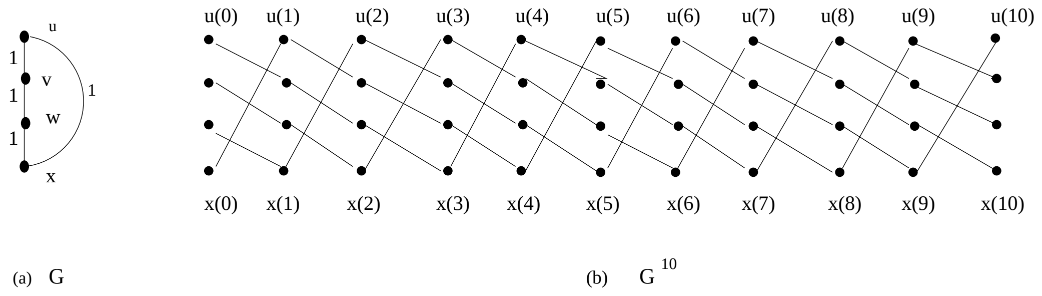
u
v
w
x
1
1
1
1
(a) G
u(0) u(1) u(2) u(3) u(4) u(5) u(6) u(7) u(8) u(9) u(10)
x(0) x(1) x(2) x(3) x(4) x(5) x(6) x(7) x(8) x(9) x(10)
(b) G 10

FIGURE 3. (a) The static graph $G(V, E)$. The 1-dimensional periodic specification is , $= (G(V, E), 1010)$. (b) The graph G^{10} specified by , .

3.5. Periodic Specifications. Next, we recall the definition of one dimensional periodic specifications due to Orlin [**Or82a**], Wanke [**Wa93**] and Höfting and Wanke [**HW95**]. For the rest of the paper $\mathbf{N}$ and $\mathbf{Z}$ are used to denote the sets of non-negative integers and of integers respectively.

3.5.1. *Periodically Specified Graphs.*

DEFINITION 3.13. *Let $G(V, E)$ (referred to as a static graph) be a finite undirected graph such that each edge (u, v) has an associated non-negative integral weight $t_{u,v}$. The two way infinite graph $G^{\mathbf{Z}}(V', E')$ is defined as follows:*

1. $V' = \{v(p) \mid v \in V$ *and* $p \in \mathbf{Z}\}$
2. $E' = \{(u(p), v(p + t_{u,v})) \mid (u, v) \in E$, $t_{u,v}$ *is the weight associated with the edge* (u, v) *and* $p \in \mathbf{Z}\}$

A 1-dimensional periodic specification , (referred to as a 1-P*-specification) is given by , $= (G(V, E))$ and specifies the graph $G^{\mathbf{Z}}(V', E')$ (referred to as the* 1-P*-specified graph).*

A 1-P*-specification , is said to be* narrow *or* 1-level restricted *(referred to as a 1-PN-specification) if* $\forall (u, v) \in E$, $t_{u,v} \in \{0, 1\}$. *This implies that* $\forall (u(p), v(q)) \in E'$, $|p - q| \leq 1$. *Similarly, a* 1-P*-specification is* k-narrow *or* k-level restricted *if* $\forall (u, v) \in E$, $t_{u,v} \in \{0, 1, \ldots k\}$.

In the remainder of this paper we refer to graphs specified by 1-PN-specifications as 1-PN-specified graphs. We note that if we replace $\mathbf{Z}$ by $\mathbf{N}$ in Definition 3.13, we obtain *one way* infinite periodically specified graphs. It is sometimes useful to imagine a narrow periodically specified graph $G^{\mathbf{Z}}$ as being obtained by placing a copy of the vertex set V at each integral point (also referred to as lattice point) on the X-axis (or the time line) and joining vertices placed on neighboring lattice points in the manner specified by the edges in E.

DEFINITION 3.14. *Let $G(V, E)$ denote a static graph. Let $G^{\mathbf{Z}}(V', E')$ denote the two way infinite* 1-PN*-specified graph as in Definition 3.13. Let $m \geq 0$ be an integer specified using binary numerals. Let $G^m(V^m, E^m)$ be the subgraph of $G^{\mathbf{Z}}(V', E')$ induced by the vertices $V^m = \{v(p) | v \in V$ and $0 \leq p \leq m\}$. A 1-dimensional finite periodic specification , (referred to as a* 1-FPN*-specification) is given by , $= (G(V, E), m)$ and specifies the graph G^m (referred to as the* 1-FPN*-specified graph).*

(It is important to observe that m is specified using binary notation.) The size of the 1-FP-specification , $= (G(V, E), m)$ (denoted by $size(,)$) is given by $size(,) = |V| + |E| + bits(m)$, where $bits(m)$ is number of bits in the numeral m. *In the rest of the paper we use m to denote both the integer and its binary representation; the intended meaning should be clear from the context.* An example of a periodic specification and the associated graph appears in Figure 3.

3.5.2. *Periodically Specified Formulas.* In Section 3.4.1, we extended the definition of satisfiability problems when instances are specified by standard specifications so as to apply to L-specified satisfiability problems. Orlin [**Or82a**] defined the problem 1-PN-3SAT; the problem 3SAT when instances are specified using 1-PN-specifications. Here we define 1-FPN- and 1-PN-specified generalized CNF satisfiability problems.

Let $U_1 = \{u_1, \ldots, u_n\}$ be a finite set of variables (referred to as static variables). Let $U = \{u_1, \ldots, u_n, \overline{u_1}, \ldots, \overline{u_n}\}$. Define the following sets.

$$\forall t \in \mathbf{Z},\ U(t) = \{u_k(t) \mid 1 \leq k \leq n\}$$
$$U^{\mathbf{Z}} = \bigcup_{t \in \mathbf{Z}} U(t)$$
$$U^{\mathbf{N}} = \bigcup_{t \in \mathbf{N}} U(t)$$
$$U^m = \bigcup_{t \in \mathbf{N},\ t \leq m} U(t)$$

(In our proofs, variable $u_k(t)$ denotes the variable u_k at time t.) If w is a literal of U, then $w(t)$, is a literal of $U^{\mathbf{Z}}$. Let $C(t, t+1)$ be a parameterized conjunction of 3 literal clauses such that each clause in $C(t, t+1)$ consists of literals from the set $U(t) \cup U(t+1)$ with the constraint that at least one literal is of from the set $U(t)$. We refer to the clauses $C(t, t+1)$ as *static* *narrow clauses.* Let $C^{\mathbf{Z}} = \bigwedge_{t \in \mathbf{Z}} C(t, t+1)$ and $C^{\mathbf{N}} = \bigwedge_{t \in \mathbf{N}} C(t, t+1)$. Given U^m and $C^{\mathbf{Z}}$, let

$C^m = \{(w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \mid$
$(w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \in C^{\mathbf{Z}}$ & $w_1(i_1), w_2(i_2), w_3(i_3) \in U^m\}$

DEFINITION 3.15. *A 1-dimensional two way infinite (finite) periodic narrow specification (denoted by* 1-PN (FPN)*-specification) of a* 3CNF *formula* $F^{\mathbf{Z}}(U^{\mathbf{Z}}, C^{\mathbf{Z}})$ $(F^m(U^m, C^m))$ *is given by* $(\, , \, = (U_1, C(t, t+1))$, $(\, , \, = (U_1, C(t, t+1), m))$, *where,* U_1 *is a a finite set of variables,* $C(t, t+1)$ *is a collection of static narrow 3 literal clauses. (In case of finite specifications* m *is a non-negative integer specified in binary.) The size of the specification denoted by* $size(\,,\,) = |U_1| + |C(t, t+1)|$. *(In case of finite specifications* $size(\,,\,) = |U_1| + |C(t, t+1)| + bits(m)$, *where* $bits(m)$ *denote the number of bits used to represent* m.*)*

The problem 1-PN-3SAT *(problem* 3SAT *when instances are specified using* 1-PN*-specifications) is the problem of determining if a* 3CNF *formula* $F^{\mathbf{Z}}(U^{\mathbf{Z}}, C^{\mathbf{Z}})$ *specified by* $\, , \, = (U_1, C(t, t+1))$ *is satisfiable.*

Similarly, the problem 1-FPN-3SAT *(problem* 3SAT *specified using* 1-FPN*-specifications) is the problem of determining if a* 3CNF *formula* $F^m(U^m, C^m)$ *specified by* $\, , \, = (U_1, C(t, t+1), m)$ *is satisfiable.*

As in the case of periodically specified graphs, it is useful to imagine a narrow periodically specified formula $G^{\mathbf{Z}}$ as being obtained by placing a copy of the variable set U at each integral point (also referred to as time unit) on the X-axis (or the time line). Furthermore, assume that the clauses $C(t, t+1)$ are placed at time t. With this notation, we can now refer to literals $U(t)$, as the set of literals at time t and the clauses $C(t, t+1)$ as the set of clauses at time t.

For each finite set S of finite arity Boolean relations, it is straightforward to extend the above definition so as to define the problems 1-PN-SAT(S) and 1-FPN-SAT(S) and hence we omit these definitions. Observe that 1-FPN-specified graphs or formulas can be exponentially larger than their input specifications. As already mentioned, we use $\mathcal{N}$ to denote the size of the formula $F^m(U^m, C^m)$ represented using standard (non-succinct) specifications.

EXAMPLE 3.16. *Let* U_1 *be the set of static variables given by* $U_1 = \{x_1, x_2, x_3\}$. U *denotes the set of literals corresponding to* U_1. *The set of static clauses are given by* $C(t, t+1) = (x_1(t) + x_2(t) + x_3(t)) \wedge (x_1(t +$

$1) + x_3(t)) \wedge (x_3(t+1) + x_2(t))$. *Let* $F = (U_1, C(t,t+1), 11)$ *be an instance of* 1-FPN-3SAT. *The formula* $F^3(U^3, C^3)$ *denoted by* , *is given by*

$$(x_1(0) + x_2(0) + x_3(0)) \wedge (x_1(1) + x_3(0)) \wedge (x_3(1) + x_2(0)) \bigwedge$$
$$(x_1(1) + x_2(1) + x_3(1)) \wedge (x_1(2) + x_3(1)) \wedge (x_3(2) + x_2(1)) \bigwedge$$
$$(x_1(2) + x_2(2) + x_3(2)) \wedge (x_1(3) + x_3(2)) \wedge (x_3(3) + x_2(2)) \bigwedge$$
$$(x_1(3) + x_2(3) + x_3(3)).\square$$

3.5.3. 1-FPN(BC)-*Specified Graphs and Formulas.* Some instances of problems arising in practice involve a periodic specification of the graph or a formula along with explicit initial and final conditions. We call such periodic specifications as periodic specifications with boundary conditions (BC). Observe that for 1-PN-specifications the concept of boundary conditions is not well defined since the expanded graph (or formula) is infinite in both directions.

Definition 3.17. *Let* $G(V,E)$ *(referred to as a static graph) be a finite undirected graph. Let* $E = E_1 \cup E_2 \cup E_3$, *where* $E_i \cap E_j = \phi$, $i \neq j$, $1 \leq i,j \leq 3$, *be a disjoint collection of edges. Each edge* $(u,v) \in E_2$ *has an associated non-negative integral weight* $t_{u,v} \in \{0,1\}$. *Let* m *be a positive integer specified using binary numerals. The 1-dimensional finite graph* $G^m(V^m, E^m)$ *is defined as follows:*

1. $V^m = \{v(p) \mid v \in V \text{ and } 0 \leq p \leq m\}$
2. $E^m = E_1^m \cup E_2^m \cup E_3^m$ *where*
   (a) $E_1^m = \{(u(0), v(0) \mid (u,v) \in E_1 \}$
   (b) $E_2^m = \{(u(p), v(p + t_{u,v})) \mid (u,v) \in E_2,\ u(p), v(p+t_{u,v}) \in V^m$, *and* $0 \leq p \leq m\}$
   (c) $E_3^m = \{(u(m), v(m)) \mid (u,v) \in E_3 \}$.

A 1-dimensional periodic specification with boundary conditions , *(referred to as* 1-FPN(BC)-*specification) is given by* , $= (G(V,E), m)$ *and specifies the graph* $G^m(V^m, E^m)$ *(referred to as* 1-FPN(BC)-*specified graph).*

A similar concept of boundary conditions can be formulated for instances of satisfiability problems. In such a case we have a static set of variables $U_1 = \{u_1, \ldots, u_n\}$ (Again U represents the associated literals). We place a copy of the variables at each time $t \in \{0, \cdots m\}$. $U(t)$ represents the copy of U at time t. There set of static clauses C consists of three disjoint sets and is given by

$$C = C_1(0) \cup C_2(t, t+1) \cup C_3(m).$$

The set of explicit clauses $C_1(0)$ contain literals from $U(0)$. $C_2(t, t+1)$ is identical to the clauses used to specify 1-FPN-formulas and contain literals from $U(t) \cup U(t+1)$. Finally we have explicit clauses $C_3(m)$ over $U(m)$. The 1-dimensional periodic specification of a 3CNF formula (referred to as 1-FPN(BC)-specification) is given by , $= (U_1, C, m)$. The 3CNF formula specified by , (referred to as 1-FPN(BC)-specified formula) is given by

$$F^m_{BC}(C^m, U^m) = C_1(0) \bigwedge C_2^m \bigwedge C_3(m)$$

Example 3.18. *Let U_1 be the set of static variables given by $U_1 = \{x_1, x_2, x_3\}$. The set of static clauses $C = C_1(0) \cup C_2(t, t+1) \cup C_3(3)$ are given by*

$$C_1(0) = (x_1(0) + \overline{x_2(0)}) \wedge (x_2(0) + \overline{x_3(0)})$$

$$C_2(t, t+1) = (x_1(t) + x_2(t) + x_3(t)) \wedge (x_1(t+1) + x_3(t)) \wedge (x_3(t+1) + x_2(t)),$$

$$C_3(3) = (x_1(3) + \overline{x_2(3)}) \wedge (x_2(3) + \overline{x_3(3)})$$

Note that $C_2(t, t+1)$ is the same as in Example 3.16. Let , $= (U_1, C, 11)$ be an instance of 1-FPN-3SAT*. The formula $F^3_{BC}(U^3, C^3)$ denoted by , is given by*

$$(x_1(0) + \overline{x_2(0)}) \wedge (x_2(0) + \overline{x_3(0)}) \bigwedge$$

$$(x_1(0) + x_2(0) + x_3(0)) \wedge (x_1(1) + x_3(0)) \wedge (x_3(1) + x_2(0)) \bigwedge$$

$$(x_1(1) + x_2(1) + x_3(1)) \wedge (x_1(2) + x_3(1)) \wedge (x_3(2) + x_2(1)) \bigwedge$$

$$(x_1(2) + x_2(2) + x_3(2)) \wedge (x_1(3) + x_3(2)) \wedge (x_3(3) + x_2(2)) \bigwedge$$

$$(x_1(3) + x_2(3) + x_3(3)) \bigwedge$$

$$(x_1(3) + \overline{x_2(3)}) \wedge (x_2(3) + \overline{x_3(3)}).\square$$

4. Translation Theorem

In this section we prove a translation theorem which allows us to relate L- and 1-FPN-specifications. The basic idea behind the theorem is simple. Intuitively, a 1-FPN-specification defines a graph in which a number of copies of a basic graph (formula) are connected in the form of a linear chain. Now imagine we had a L-specification G_1 for specifying roughly half of the object. The complete L-specification of the expanded object will have two calls to G_1 and a subgraph that will connect the objects represented by each of the modules by a using copies of the basic module.

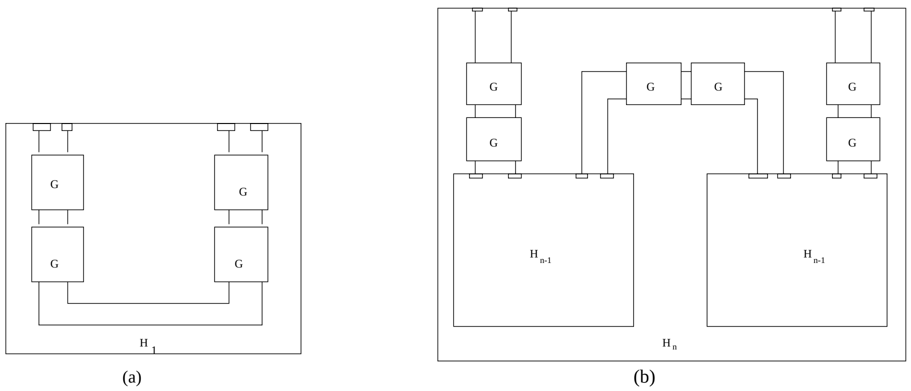

FIGURE 4. Construction of H_i, $1 \le i \le n$. In the figure, each of the $|l_i|, |r_i|$ and $|m_i|$ are 2 and H_1 consists of 4 copies of G.

THEOREM 4.1. *There is a polynomial time transformation, that maps a* 1-FPN(BC)*-specification* , $= (G(V,E), m)$ *of the graph* G^m *to a 1-level-restricted* L*-specification* , $_1$ *of a graph* $E(,\ _1)$*, such that the graphs* G^m *and* $E(,\ _1)$ *are isomorphic and* $size(,\ _1) = O((size(,\ ))^2)$.

Proof: We discuss the transformation without boundary conditions first. Given a 1-FPN-specification , $= (G(V,E), m)$ of G^m we construct a L-specification $H_G = (H_1, \cdots H_n)$ as follows. The non-terminal H_1 consists of at least 2 copies of G (i.e. G^2). They are connected in series as shown in Figure 4(a).

Non-terminal H_i**,** $2 \le i \le n$**:** We show how to specify the graph G^k by the L-specification $(H_1, \ldots, H_i)$. The non-terminal H_i consists of five components l_i, r_i, m_i, H_{i-1}, H_{i-1}. Each of l_i, r_i and m_i consists of at least 1 copy of G (G^2) attached in a manner as shown in the Figure 4(b). Each H_{i-1} recursively encodes G^p, where p will be specified later. The H_{i-1}'s are connected to the components l_i, r_i and m_i as shown in Figure 4(b). The sizes of l_i, r_i and m_i and p should satisfy the following constraints.

$$|l_i|,\ |r_i|,\ |m_i| \ge 1$$

$$p + p + |l_i| + |r_i| + |m_i| = k$$

The first constraint is needed so that the resulting specification is 1-level-restricted. The second equation says that the total number of copies of G is no more than k, which is the size of the graph we want to encode. It is clear that the sizes of l_i, r_i and m_i can be chosen so that the above equations can be satisfied. The process can be carried out recursively to obtain the required specification $(H_1, \ldots H_i)$ in polynomial time.

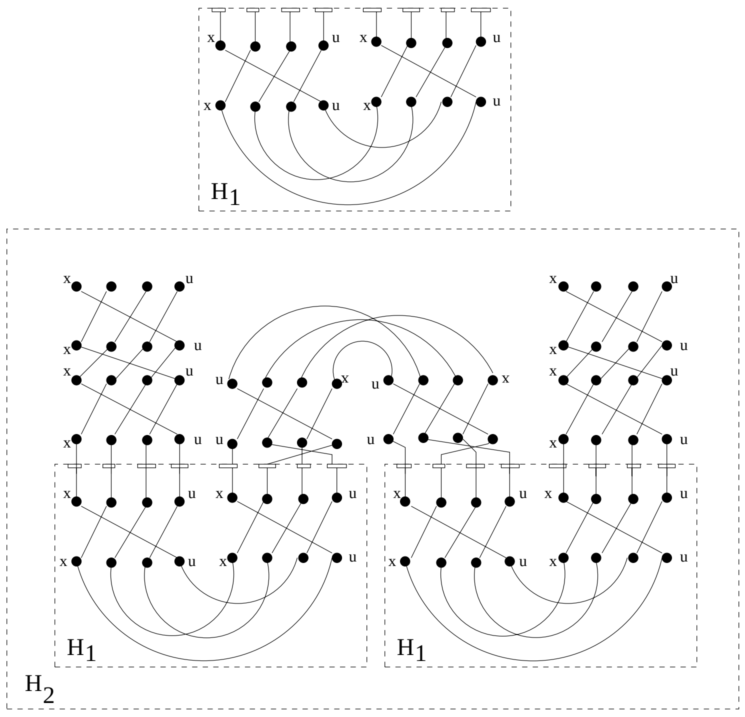

FIGURE 5. An L-specification , $= (H_1, H_2)$ of the graph G^{10} represented by the static graph G depicted in Figure 3. Observe that the graphs G^{10} and $E(,\ )$ are isomorphic.

Observe that a similar transformation can be carried out if we start with a 1-FPN(BC)-specified graph G. In this case the graphs specifying the boundary conditions are included in the highest non-terminal of the L-specification constructed.□

Using similar ideas we can transform 1-FPN-specified formulas into isomorphic L-specified formulas. Thus we have the following corollary.

COROLLARY 4.1.1. *There is a polynomial time transformation that maps a* 1-FPN(BC)*-specification* , $= (F(U_1, C(t, t+1)), m)$ *of the formula* F^m *to a 1-level-restricted* L*-specification* $,_1$ *such that that the formulas* F^m *and* $E(,_1)$ *are isomorphic and* $size(,_1) = O((size(,\ ))^2)$.

EXAMPLE 4.2. *Consider the static graph in Figure 3. Figure 5 shows the* L*-specification of the graph* G^{10}*. Again, the one-to-one correspondence between the pins and their neighbors is clear from the figure and hence is not given explicitly. For the sake of exposition we have depicted the graphs* H_1 *in the definition of* H_2.

5. PSPACE-completeness of α-3SAT

5.1. Space bounded reductions. We assume that the reader is familiar with log-space reductions and polynomial time reductions ($\leq_p$). Let us say that $A \subseteq \ ,^*$ is PSPACE-complete **via a size** $L(n)$ reduction, if A is in PSPACE and for any $B \subseteq \Delta^*$ such that B is decidable (non)-deterministically in space $S(n) = n$, $B \leq_p A$ via some function f which can be computed deterministically in polynomial time, such that for all $w \in \Delta^*$, $|f(w)| \leq cL(|w|)$, where c is a constant depending on B. If $f(n) = O(n)$, we say that the reduction is a linear size reduction. If $f(n) = O(n(\log n)^k)$ then we call the reduction a *quasi-linear size* reduction.

Most of the problems proved hard in this paper are by $O(n \log n)$ size reductions; moreover these reductions are realized by functions f such that the number of symbols used are only $O(n)$. (In other words the $\log n$ factor arises only from the need to write out the names of the variables.) This fact can be easily verified from the proofs of hardness given in this paper.

THEOREM 5.1. 1. 1-FPN-3SAT *is in* NSPACE(n).
2. *There is a* $\Theta(n \log n)$ *size quasi-linear time reduction from the membership problem for an arbitrary non-deterministic linear space bounded machine (LBA) to the problem* 1-FPN-3SAT. *Thus* 1-FPN-3SAT *is* PSPACE*-complete.*

Proof:
Part (1): We first show that the problem is in NSPACE(n). Consider an LBA M. Given a 1-FPN-specification $\ , = (F(U_1, C(t, t+1)), m)$ of the formula F^m, M needs to maintain at any given time t only assignments to variables at time t and time $t+1$. Hence M can verify that the instance of 1-FPN-3SAT is satisfiable as follows. At each step t it guesses an assignment to the variables at time $t+1$. It also remembers the assignment to the variables at time t. Using these values it verifies that the clauses at time t are indeed satisfied. The maximum number of time steps $(2^m + 1)$ can be kept track of using a counter of size $O(m)$ (since m is specified as a binary numeral). This proves that given an instance , of 1-FPN-3SAT, we can recognize in non-deterministic *linear* space (i.e in space $O(size(,))$) if F^m is satisfiable.

Part (2): Next, we prove 1-FPN-3SAT is PSPACE-hard. Given a non-deterministic LBA, M, with input x (where $|x| = n$), we construct an instance of 1-FPN-3SAT $\ , = (F_x(U_1, C(t, t+1)), m)$ such that $size(F_x) =$

$O(n)$, and x is accepted by M if and only if F_x^m is satisfiable. The reduction consists of two phases.

Phase 1: In the first phase, we start with the given LBA M with input $x = (x_1, \ldots, x_n)$ and construct a new LBA M_1 which simulates M on x with the following additional properties that

1. if the LBA M does not accept x then each computation of M_1 on x halts within $2^{c_0 n}$ moves, and
2. if the LBA M accepts x then M_1 has a cycling computation, where the length of an ID never exceeds $O(|x|)$.

M_1 can be constructed easily by adding an auxiliary clock to serve as a counter. M_1 now just simulates M. If M enters a final configuration, then M_1 repeats this configuration. It is clear that M_1 accepts x if and only if M accepts x. Moreover it is easy to see that M_1 has the two desired properties above.

Phase 2: The second phase consists of constructing an instance $, = (F_x(U_1, C(t, t+1)), m)$ of 1-FPN-3SAT by a polynomial time reduction from M_1. Now we know that each ID of the Turing machine M_1 is of length $O(n)$, where n is the size of the input. Since M_1 is a non-deterministic LBA we need to consider only 2^{Dn} different ID's for our reduction. (Here D is an appropriately chosen constant.) We can choose an encoding of states and symbols of M_1 into words in $\{0,1\}^*$ so that every ID of M_1 will consist of $c_1 n$ Boolean variables where $n = |x|$ and c_1 is a constant independent of x. Let $\mathcal{U}_1$ denote this set of Boolean variables. Let $ID(t)$ denote the ID of the Turing machine at time t. This is specified by appropriate values to the variables $\mathcal{U}_1(t)$. We also have a set of $Dn + 1$ Boolean variables encoding a counter $\gamma(t)$. The counter values range from 0 to 2^{Dn}. $C(t, t+1) = f_1(t, t+1) \wedge f_2(t, t+1) \wedge f_3(t, t+1)$. We discuss each of the three formulas $f_i(t, t+1)$, $1 \le i \le 3$.

1. The formula $f_1(t, t+1)$ encodes a counter which is given by

$$\gamma(t+1) = (\gamma(t) + 1) \pmod{2^{Dn} + 1}.$$

The intended meaning of the equation is that the counter resets to 0 after every $2^{Dn} + 1$ time units. It is easy to see that the counter can be simulated by a CNF formula, in which each clause has variables that are no more than one time unit apart. We briefly discuss how to simulate a counter here. Let $q = Dn$. We use Boolean variables $d_q, d_{q-1}, \ldots, d_0$ to simulate a counter. The variable d_0 encodes the lowest order bit and the bit d_q denotes the highest order bit. We also use Boolean variables $c_q, c_{q-1}, \ldots c_0$ to keep track of the carry bits required to do the addition. Let $d_i(t)$ denote the copy of the variable d_i at time unit t. Let $\mathcal{U}_2 = \{d_q, d_{q-1}, \ldots d_0, c_q, c_{q-1}, \ldots, c_0\}$ The formula f_1 is expressed as follows:

$$f_1(t, t+1) \equiv (g_1 \Rightarrow h_1) \wedge (\overline{g_1} \Rightarrow h_2).$$

We describe each of the subformulas g_1, h_1 and h_2 in the following. g_1 checks to see if the counter needs to be reset. If g_1 is true then h_1 merely resets the counter; else h_2 increments it by one. Hence g_1 is given by

$$g_1 \equiv [(d_0(t) \wedge d_1(t) \wedge \ldots \wedge d_q(t))]$$

The condition that counter resets to 0 after 2^{Dn} time units can now be written using a CNF formula h_1 as follows:

$$h_1 \equiv \left[(\overline{d_0(t+1)} \wedge \overline{d_1(t+1)} \wedge \ldots \wedge \overline{d_q(t+1)})\right].$$

As mentioned earlier, we have $q+1$ carry bits to do the addition. Let the carry bits corresponding to the counter $\gamma(t)$ be $c_q, \ldots c_0$. h_2 is now defined as a conjunction of the following clauses

$$h_2 \equiv \left(d_0(t+1) = \overline{d_0(t)}\right) \wedge$$

$$\bigwedge_{i=1}^{i=q} \left(d_i(t+1) = [\overline{d_i(t)} \wedge c_i(t+1)] + [d_i(t) \wedge \overline{c_i(t+1)}]\right)$$

$$\bigwedge (c_0(t+1) = d_0(t))$$

$$\wedge \bigwedge_{i=1}^{i=q} (c_{i+1}(t+1) = d_i(t) \wedge c_i(t+1))$$

Observe that we have $O(n)$ Boolean variables encoding the counter. Therefore, the size of each of the formulas g_1, h_1 and h_2 is linear in the size of the input. Furthermore each of the formulas g_1, h_1 and h_2 contain a linear (in the number of Boolean variables used to simulate the counter) number of clauses. As a result, the implications $(g_1 \Rightarrow h_1)$ and $(\overline{g_1} \Rightarrow h_2)$ can be written in equivalent 3CNF form using additional temporary variables. The size of the 3CNF formulas is linear in size of the implications. Therefore $size(f_1(t, t+1)) = O(n)$. Let $\mathcal{U}_3$ denote the set of temporary variables used to transform the formula to an equivalent 3CNF formula.

2. The formula $f_2(t, t+1)$ enforces the condition that when the counter value is 0, the variables $\mathcal{U}_1$ encode the starting ID of the Turing machine. Let $start(ID(t))$ denote a 3CNF formula that checks if $ID(t)$ is the initial ID of the machine. Thus $f_2(t, t+1)$ is a 3CNF formula that encodes the implication $((\gamma(t) = 0) \Rightarrow start(ID(t)))$. Again, we can verify that $f_2(t, t+1)$ can be written as a 3CNF formula in polynomial time. By standard arguments it follows that $size(f_2(t, t+1))$ is $O(n)$.
3. The formula $f_3(t, t+1)$ is needed to ensure that starting at the second ID, each subsequent ID of M_1 follows from the previous ID by using the transition function of M_1. (Recall that the notation $(X \vdash_M^j Y)$ means that machine M, starting with ID X, can produce the ID Y in exactly j steps.) Thus $f_3(t, t+1)$ is a 3CNF formula encoding the following implication.
$$(1 \leq \gamma(t) \leq 2^{Dn}) \Rightarrow (ID(t) \vdash_M ID(t+1))$$
The function $(ID(t) \vdash_M ID(t+1))$ can be expressed by a 3CNF formula whose size is linear in n as shown in [**Hu73a**]. Moreover the 3CNF formula depends on the current value of the counter. $f_3(t, t+1)$ is a narrow 3CNF formula since the clauses at time t contain variables only at times t and $t+1$.

Let $m = 2^{2Dn}$. The static variables U_1 are given by $U_1 = \mathcal{U}_1 \cup \mathcal{U}_2 \cup \mathcal{U}_3$. As before U denotes the corresponding set of literals. $C(t, t+1)$ is the conjunction of clauses obtained after carrying out Steps 1 to 3 above. This completes the description of the instance $\Gamma = (F_x(U_1, C(t, t+1)), m)$ of 1-FPN-3SAT. Γ represents the 3CNF formula $F_x^m = \bigwedge_{t=0}^{m} C(t, t+1)$.

We now prove the correctness of our reduction, i.e. we prove that F_x^m is satisfiable if and only if M accepts x. If the Turing machine M accepts x then we know that M_1 has a cycling computation. Hence by setting $d_t = 0$, we can ensure that $f_2(0, 1)$ is satisfied. Now the remaining variables can be assigned appropriate values so as to satisfy the formula F_x^m. Conversely, assume that F_x^m is satisfiable. Since D (and hence m) are suitably large integers, it is guaranteed that the simulation must be carried out for enough steps so that the Turing machine M_1 goes through the sequence $d_t = 0, d_t = 1, d_t = 2, \cdots d_t = 2^{Dn}$. This implies that the formulas $f_2(t, t+1)$ and $f_3(t, t+1)$ would be true from then on and therefore the Turing machine M accepts x.$\square$

Corollary 5.1.1.

1. *The problem* 1-FPN(BC)-3SAT *is in* NSPACE(n)*. There is a quasi-linear time, quasi-linear size reduction from the membership problem for an arbitrary non-deterministic* LBA *to* 1-FPN(BC)-3SAT*. Thus the problem* 1-FPN(BC)-3SAT *is* PSPACE*-complete.*
2. *The problem* L-3SAT *is in* DSPACE(n)*. There is a $O(n^2)$ time and $O(n^2 \log n)$ size reduction from the membership problem for an arbitrary non-deterministic* LBA *to* L-3SAT*. Thus the problem* L-3SAT *is* PSPACE*-complete*

Proof: The PSPACE-hardness of 1-FPN(BC)-3SAT follows immediately from Theorem 5.1. By arguments similar to those presented in the proof of Theorem 5.1, the problem 1-FPN(BC)-3SAT is seen to be in NSPACE(n).

The PSPACE-hardness of the problem L-3SAT follows from Theorem 5.1 and the Translation theorem. Specifically, there is a $O(n^2)$ size reduction from the membership problem for a non-deterministic LBA to the problem L-3SAT. We now show that L-3SAT is solvable in DSPACE(n) by using backtracking. Let $F_i(X^i) = (\bigwedge_{1 \le j \le l_i} F_{i_j}(X^i_j, Z^i_j)) \bigwedge f_i(X^i, Z^i)$. Here $0 \le i_j < i$. Given an assignment of X^i, we evaluate $F_i(X^i)$ as follows: For each assignment of Z^i: For $1 \le j \le l_i$, check if $F_{i_j}(X^i_j, Z^i_j)$ is true and store the values of $F_{i_j}(X^i_j, Z^i_j)$. These values require space linear in l_i. Finally evaluate $f_i(X^i, Z^i)$. This evaluation can be carried out in deterministic space $O(size(f_i))$ since f_i is a CNF formula. If no such assignment to Z^i is found, $F_i(X^i)$ is false. Otherwise, $F_i(X^i)$ is true. Thus, if F_k for $k < i$, can be solved in space linear in their representation, F_i can be solved in space linear in its representation, since the space needed to store F_i is the space to store the values of X^i, Z^i and the intermediate results $F_{i_j}(X^i_j, Z^i_j)$. This allows us to do the backtracking in DSPACE(n). F_1 can be solved in DSPACE(n) since it is a CNF formula. Hence by induction on the number of levels in the hierarchy, membership of the problem L-3SAT in DSPACE(n) follows.□

Thus the results yields nearly tight lower and upper bound (under certain complexity theoretic assumptions) on the space required for solving the problem L-3SAT.

Corollary 5.1.2. *The problem* 1-PN-3SAT *is in* NSPACE(n)*. There is a quasi-linear time, quasi-linear size reduction from the membership problem for an arbitrary non-deterministic* LBA *to* 1-FPN(BC)-3SAT*. Thus the problem* 1-PN-3SAT *is* PSPACE*-complete.*

Proof: Orlin [**Or82a**] and Papadimitriou [**Pa94**] have shown that the problem 1-PN-3SAT is in NSPACE(n). We only prove that the problem is PSPACE-hard by a $O(n \log n)$ size reduction from the membership problem for a non-deterministic LBA. (In contrast, the reduction in [**Or82a, Pa94**] is an $O(n^2 \log n)$ size reduction.) Observe that if an instance , $=$ $(F(U_1, C(t, t+1)))$ of 1-PN-3SAT has a solution then the assignment to the variables is periodic. Specifically, if there are n variables in the static formula we need to consider only 2^{2n} distinct assignments to the variables in U_1. Thus, starting from , we can construct an instance $(F(U_1, C(t, t+1)), m)$ of 1-FPN-3SAT (where $n = |U_1|$ and $m = 2^{2n}$) such that F^m is satisfiable if and only if $F^{\mathbf{Z}}$ is satisfiable. This completes the proof that the problem 1-PN-3SAT is PSPACE-complete.□

Remark 1: The basic technique underlying the proof of PSPACE-hardness of 1-FPN-3SAT is the ability to use a counter to implicitly make sure that there is a sequence of formulas which check if the TM starts right.

Remark 2: The above results illustrate the basic ideas underlying the PSPACE-hardness results in both Lengauer and Wagner [**LW92**] and in Orlin [**Or82a**]. Specifically, our results show that the ability to represent $2^{\Theta(n)}$ simple repetitive structures of size $\Theta(n \log n)$ by specifications of size $O(n \log n)$ can make the problems PSPACE-hard and such repetitive structures can be represented by 1-PN- as well as L-specifications.

As our next corollary shows, 1-FPN-3SAT is PSPACE-hard even when restricted to formulas with bandwidth $O(\log \mathcal{N})$. Recall that $\mathcal{N}$ denotes the size of the expanded formula.

Corollary 5.1.3. *There is a quasi-linear time, quasi-linear size reduction from the membership problem for an arbitrary non-deterministic* LBA *to* 1-FPN-3SAT *even when restricted to formulas with bandwidth* $O(\log \mathcal{N})$*. Thus the problem* 1-FPN-3SAT *is* PSPACE*-complete even when restricted to formulas with bandwidth* $O(\log \mathcal{N})$*.*

Proof: The proof follows by observing that the following numbering scheme to the variables of the formula obtained in Theorem 5.1 yields a $O(\log \mathcal{N})$ bandwidth layout. Let $C(t, t+1)$ have p clauses. Then number the clauses at time t using numbers from pt to $(p+1)t$. The numbering of clauses at a given time t can be carried out in any arbitrary order.□

Remark 3: As shown in Section 6, Corollary 5.1.3 in conjunction with local replacement type reductions between problems specified using standard specifications can be used to prove that several classical graph problems are

PSPACE-hard even for $O(\log \mathcal{N})$ bandwidth bounded graphs specified using *either* 1-FPN-specifications or 1-level-restricted L-specifications (since the L-specification obtained by the translation theorem represents an isomorphic graph (formula)).

Next, we discuss the PSPACE-hardness of the problems 1-FPN(BC)-3SATWP and 1-FPN(BC)-3SATWP. Recall that, 1-FPN-3SATWP (problem 3SATWP specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^m(U^m, C^m)$ containing at most one negated literal per clause specified by , $= (F(U_1, C(t, t+1)), m)$ is satisfiable. In [**MH+96**], we show that

THEOREM 5.2. *There is a quasi-linear time and quasi-linear size reduction from the membership problem for an arbitrary deterministic* LBA *to the problems* 1-FPN(BC)-3SATWN *and* 1-FPN(BC)-3SATWP. *Thus the problems* 1-FPN(BC)-3SATWN *and* 1-FPN(BC)-3SATWP *are* PSPACE-*complete.*

Several remarks are in order at this point. We make these remarks with respect to the problems α-3SATWN. Similar remarks hold for α-3SATWP.

1. While 1-FPN-3SAT is NSPACE(n)-hard, 1-FPN(BC)-3SATWN is only shown to be DSPACE(n)-hard.
2. While 1-FPN(BC)-3SATWN DSPACE(n)-hard, the problem 1-FPN-3SATWN is polynomially decidable. This points out an important difference between 1-FPN- and 1-FPN(BC)-specifications.

6. Applications

In this section, we briefly discuss how our hardness results for CNF satisfiability problems can be applied to prove hardness results for several combinatorial problem when specified using one of the specifications α. These problems have already been used to prove various other hardness results obtain in [**MR+93, MH+93a**].

6.1. Basic Technique.

In [**LW92**], Lengauer and Wagner proved the PSPACE-hardness of several L-specified graph problems by a polynomial time reduction from QBF [**GJ79**]. Our approach consists of reductions from the problems L-3SAT and 1-FPN-3SAT. The basic idea behind all our reductions is illustrated by the following example given in [**MH+93a, MR+93**].

Consider the INDEPENDENT SET (IS) problem for planar graphs. When the input instance (in this case, the input graph) is specified using one of the standard representations, the problem can be shown to be NP-hard by a *local* *reduction* from 3SAT [**Li82**]. Roughly speaking, the phrase "local reduction" refers to a reduction where each clause and each variable is replaced by a *fixed size* subgraph or gadget.

The problem L-PI-IS for L-specified planar graphs can be shown to be PSPACE-hard by a polynomial time reduction from L-PI-3SAT. Such a reduction proceeds bottom up and corresponding to each non-terminal F_i in the specification $F = (F_1, \dots F_n)$ of L-3SAT, creates a non-terminal G_i. Thus, $G = (G_1, \dots, G_n)$ represents the hierarchical specification obtained at the end of the transformation. The crucial property of such a reduction is that $E(G)$ is exactly the **same** graph as would be obtained by carrying out the reduction in [**Li82**] on $E(F)$ laid out in the plane in a certain fashion. This observation leads directly to the PSPACE-hardness of L-PI-IS for L-specified graphs.

We call this process of transforming the given instance "level by level", **lifting** the reduction. Our idea then, is to lift the known reduction from 3SAT to problem Π when the instance is specified non-hierarchically, and thus obtain a suitable reduction from L-3SAT to the problem L-Π. This approach helps one to prove easily many more PSPACE-hardness results for hierarchically specified instances (see Table 1). We also point out that most of the reductions are quasi-linear size and quasi-linear time reductions and thus provide tight lower bounds on the deterministic time complexity of the problem (under standard complexity theoretic assumptions).

A similar approach can be taken for proving PSPACE-hardness of 1-FPN specified problems. Consider once again the problem IS. Starting from the set of static clauses that specify an instance of 1-FPN-3SAT, we construct a static graph by performing the known local reductions from 3SAT to IS. This gives us an instance of the problem 1-FPN-IS such that the expanded graph has an independent set of a given size if and only if the expanded formula is satisfiable. Problems such as INDEPENDENT SET, DOMINATING SET, CLIQUE COVER and 3-COLORING have been proved to be NP-hard for planar graphs using such local reductions [**Li82, GJ79**].

Note that if we prove a graph problem to be PSPACE-hard for 1-FPN or 1-FPN(BC)-specified instances then the hardness result for L-specified instances follows directly from the translation theorem.

6.2. Planar Satisfiability and Graph Problems. We first show that the problem 1-FPN-Pl-3SAT is PSPACE-complete. To do this, we introduce some additional notation. Let , $= F(U, C(t, t+1), m)$ be an instance of 1-FPN-3SAT (i.e. the static formula), where $U = \{u_1, \dots, u_n\}$ and $C(t, t+1) = \{c_1, \dots, c_q\}$. Define an **auxiliary graph** $G(V, E)$ associated with F as follows: $V = V(1) \cup V(2) \cup CV$, where $V(1)$ and $V(2)$ corresponds to two disjoint copies of the variables U and the vertices in CV are in one-to-one correspondence with $C(t, t+1)$. Thus for each variable $w_k \in U$ we have two vertices $v_k(1)$ and $v_k(2)$ where $v_k(1) \in V(1)$ and $v_k(2) \in V(2)$. Let $c_j = (w_1(i_1) \vee w_2(i_2) \vee w_3(i_3))$. $i_1, i_2, i_3 \in \{t, t+1\}$. Then $E(j) = \{(cv_j, v_1(i_1 - t + 1)),\ (cv_j, v_2(i_2 - t + 1)),\ (cv_j, v_3(i_3 - t + 1))\}$. $E = \bigcup_{j=1}^{j=q} E(j)$. A **nice layout** $\mathcal{L}_G$ of the auxiliary graph $G(V, E)$ is constructed as follows:

1. For $1 \le i \le n$, place $v_i(1)$ at $(0, i)$ and place $v_i(2)$ at $(2, i)$. Place the clause vertex cv_j at $(1, j)$.
2. Draw the edges in $E(j)$ so that they lie within the box $(0, 0), (2, q + n)$.

It is easy to extend the concept of nice layout to apply to graphs corresponding to $BG(F^m)$ as follows. There is one vertex for each variable and each clause in F^m. The vertex corresponding to the variable $v_i(t) \in V(t)$ is placed at $(2t, i)$. The clause vertex corresponding to a clause $cv_j(t, t+1))$ at time t (i.e. $C(t, t+1)$) is placed at $(t+1, j)$. The edges between a clause vertex at time t and a variable vertex at time t or $t+1$ is drawn to lie within the box of size $(2, q+n)$ placed with its lower left corner at $(t, 0)$. The nice layout corresponding to the graph $BG(F^m)$ is denoted by $\mathcal{L}_{F^m}$. See Figure 6 for an example. For the rest of the section we do not distinguish between a variable and its associated vertex and the intended meaning will be clear from the context.

THEOREM 6.1. 1-FPN-Pl-3SAT *is* PSPACE-*complete.*

Proof: We describe the proof in two parts. In the first part, we describe our transformation from an instance of 1-FPN-3SAT, to an instance of 1-FPN-Pl-3SAT. In the second part we prove the correctness of our transformation.
Transformation: Given an instance $(F(U, C(t, t+1), m)$ of 1-FPN-3SAT, we obtain an instance $(F_1(U_1, C_1(t, t+1), m)$ of 1-FPN-Pl-3SAT as follows.

1. Compute a nice layout $\mathcal{L}_G$ of the auxiliary graph $G(V, E)$.

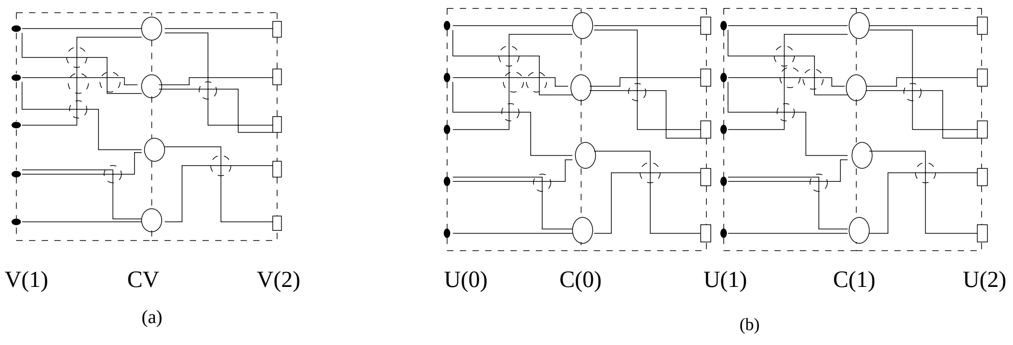

FIGURE 6. Schematic Diagram showing how to obtain an instance of 1-FPN-PI-3SAT. (a) Schematic diagram of the auxiliary graph. The black dots represent the variables and the ellipses represent the clauses. The dotted circles denote the crossovers which are replaced by crossover boxes. The squares denote the vertices $V(2)$. The dotted cycle denotes the bounding box in which all the clauses and auxiliary variables introduced in Step 2 of proof of Theorem 6.1 are placed. (b) Part of the formula F_1^m, represented by the static formula F_1 constructed in the proof of Theorem 6.1. In the figure, we have left the two copies of the auxiliary graph separately to illustrate how the layout of F_1^m is obtained starting from a layout of the auxiliary graph associated with F_1. The small squares (representing $V(2)$ in the first auxiliary graph are identified in a one-one fashion with the black dots (vertices $V(1)$) in the second auxiliary graph.

2. Note that the vertices in G correspond to the variables and clauses in F. let $R_1, \ldots R_t$, denote the set of crossovers in $\mathcal{L}_G$. Each crossover R_j is replaced by using Lichtenstein's planar crossoverbox [**Li82**], using the auxiliary set of variables $A_j(t) = \{a_j^1(t), \ldots a_j^{r_j}(t)\}$. For each set $A_j(t)$ we have the corresponding set $A_j = \{a_j^1, \ldots a_j^{r_j}\}$. Note that $A_j(t) \cap A_k(t) = A_j \cap A_k = \phi$, $k \neq j$. Let $A(t) = \bigcup_{j=1}^{j=t} A_j(t)$ and $A = \bigcup_{i=1}^{j=t} A_j$. Denote the planar graph obtained as a result of transformation by G_1.
3. Let $C_1(t, t+1)$ be the set of clauses over $A(t) \cup U(t) \cup U(t+1)$ that are obtained as a result of Step 2. $U_1 = U \cup A$. Note that this includes new clauses, the old clauses in $C(t, t+1)$ as well as clauses in $C(t, t+1)$ modified as a result replacing the crossovers.

4. $F_1(U_1, C_1(t, t+1))$ is the static formula resulting by the above transformation.

Correctness: The proof of correctness consists of two parts. In the first part we argue that the formula F_1^m is planar. In the second part, we argue that F_1^m is satisfiable if and only if F^m is satisfiable.

We obtain a layout of the formula F_1^m as follows. We create $m+1$ copies of the graph (formula) G_1 obtained at the end of Step 2 of the transformation. Let the copies be denoted by $G_1(0), \ldots, G_1(m)$. Now "glue" the copies of the these graphs by identifying the vertices $V(2)$ in $G_1(t)$ with the vertices $V(1)$ in $G_1(t+1)$ in a one-to-one fashion as depicted in Figure 6. It is easy to see that the resulting bipartite graph represents F_1^m. Thus the formula F_1^m is planar.

We prove the second part as follows. Compute a nice layout $\mathcal{L}_{F^m}$ of $BG(F^m)$. By observing Figure 6, we get that F_1^m is exactly the same formula as would be obtained by carrying out Lichtenstein's reduction on the formula F^m laid out as $\mathcal{L}_{F^m}$.

Thus F_1^m is planar and is satisfiable if and only if F^m is satisfiable. This completes the proof of the theorem.□

6.3. Application to graph problems. As another example, we show the 3-COLORING problem is PSPACE-complete for succinctly specified inputs. For this we need the following facts.

1. There is a local replacement with enforcer type reduction from NAE-3SAT to 3-COLORING [**Pa94**].
2. There is a planar crossover box for 3-COLORING (see [**GJ79**]).
3. There is a local reduction from 1-FPN-3SAT to 1-FPN-NAE-3SAT and hence 1-FPN-NAE-3SAT is PSPACE-complete.

We describe the basic construction informally and leave the details to the reader. Starting with an instance of 1-FPN-NAE-3SAT, we first show that 1-FPN-3-COLORING is PSPACE-complete. The reduction is similar to the one used for proving the PSPACE-hardness of 1-FPN-PI-3SAT. The only detail we need to explain is the use of enforcer node. This is easily taken care of by replacing a single node enforcer by a chain of triangles. The idea has been described in [**Pa94, MR+93**] and we omit the discussion here.

Similar results can be obtained for several other problems. Thus we have the following theorem.

THEOREM 6.2. *The following problems are* PSPACE*-hard for* 1-FPN- *or* L*-specified planar graphs of* $O(\log \mathcal{N})$ *bandwidth:* 3-COLORING, INDEPENDENT SET(IS), DOMINATING SET, VERTEX COVER, PARTITION INTO TRIANGLES *and* HAMILTONIAN PATH(HP).

Table 1 contains a sample of the results we have obtained for succinctly specified problems. As another example of the applicability of our results we consider the complexity of the monotone circuit value problem for L- and 1-FPN(BC)-specified inputs. We assume that the reader is familiar with the definition of the *Circuit Value problem* (CVP) [**MH+93a**]. In [**MH+96**], we show that

THEOREM 6.3. *There is a linear size reduction from the membership problem for an arbitrary deterministic linear bounded automata to the problem* 1-FPN(BC)-MCVP*. Thus* 1-FPN(BC)-MCVP *is* PSPACE*-complete. Moreover the result holds even when the expanded circuits are* $O(\log \mathcal{N})$ *bandwidth bounded.*

THEOREM 6.4. *The problems* L-LP *feasibility and* 1-FPN(BC)-LP *feasibility problems are* PSPACE*-hard.*

The result is significant since it points out that linear programming techniques for solving non-succinctly specified problems can not be directly used to solve the corresponding succinctly specified problems. In fact, we show in [**MH+96**] that even finding an approximately optimal solution for linear programming problem is PSPACE-hard.

7. Conclusions

7.1. Summary of results. We proved a translation theorem that allowed us to relate the L- and 1-FPN-specifications. We then proved the PSPACE-hardness of several basic CNF satisfiability problems for succinctly specified inputs. These results in conjunction with our translation theorem and known local reduction, provide alternative and unified PSPACE-hardness proofs for several L- or PN-specified problems considered in [**LW92, Or82a**]. We also outlined how our techniques could be used to prove the PSPACE-hardness of several other L-, 1-FPN-, 1-FPN(BC)- or 1-PN-specified problems. Table 1 contains example of the results for L-, 1-FPN-, 1-FPN(BC)- or 1-PN-specified problems that can be obtained by the techniques presented here. In the following, we explain the various entries in Table 1 and also give some related remarks.

1. An entry 1 and 4 in Table 1 denotes that the corresponding problem is PSPACE-hard, even when restricted to L- or 1-PN-specifications of $O(\log \mathcal{N})$ bandwidth bounded instances. These results have previously not appeared in the literature; 4 implies that the result is proved formally in [**MH+96**].
2. An entry 2 in Table 1 denotes that the problem L-Π or the problem 1-PN-Π was shown to be PSPACE-hard in [**LW92**] or [**Or82a**]. This can also be shown by the ideas and techniques of this paper, **even** when restricted to L and 1-PN-specifications of $O(\log \mathcal{N})$ bandwidth bounded instances.
3. An entry 3 in Table 1 denotes that the corresponding problem is polynomial time solvable. This is shown in [**MH+96**].
4. All the problems *except* Problem no 2, some of the Problems in 3, and Problem 15 remain PSPACE-hard even when restricted to **planar** $O(\log \mathcal{N})$ bandwidth bounded instances.
5. Most of the problems α-Π above are shown to be PSPACE-hard by *lifting* the quasi-linear size local reduction from 3SAT to Π given in [**HMS94, GJ79**].

7.2. Recent Related Work. As shown in several of our recent papers, periodically specified satisfiability problems are useful in proving additional results for succinctly specified problems. We mention a few of these results in the following.

1. In a companion paper [**MH+96**], we show that several natural satisfiability problems are polynomial time solvable for succinctly specified inputs. The easiness results along with the hardness results presented here are used to characterize completely the complexity of the problems α-SAT(S).
2. In [**MH+95b**], we characterize the complexities of several combinatorial and basic generalized CNF satisfiability problems SAT(S) [**Sc78**], when instances are specified using various kinds of 2-dimensional periodic specifications [**Or82a, Wa93, HW94, HW95, CM91, CM93**]. We show that these problems become NEXPTIME-hard, EXPSPACE-hard, etc, when represented by various kinds of 2-dimensional periodic specifications.
3. In [**MH+93a, MH+94, MH+95a**], we obtain polynomial time approximation algorithms with constant performance guarantees for

PSPACE-hard and NEXPTIME-hard optimization problems for hierarchically and periodically specified inputs. These include optimization versions of generalized CNF satisfiability problems as well as several graph problems specified hierarchically or periodically.

4. In [**MH+95c**], we show that periodically specified versions of the problems 3SATWP and 3SAT are efficiently reducible to a number of problems Π when instances are specified succinctly as in [**Ga82, GW83, PY86, BOW83, HLW92**]. This yields unified proofs of the hardness of the problems Π when so specified [**MH+95c**].

No	Problem Π	1-FPN -Π	L-Π	1-PN -Π	1-FPN (BC)-Π
1	3SAT,1-3SAT,MAX-2SAT	1	1	1	1
2	NAE-3SAT	1	1	1	1
3	Each of the NP-hard NP-hard SAT(S) problems in [**Sc78**]	4	4	4	4
4	0-1 INTEGER PROGRAMMING	1	1	1	1
5	LINEAR PROGRAMMING FEASIBILITY	1	1	1	1
6	SET PACKING, SET COVERING, X3C, 3DM	1	1	1	1
7	VERTEX COVER, IS	1	2	2	1
8	FEEDBACK VERTEX SET	1	1	1	1
9	FEEDBACK ARC SET	1	1	1	1
10	HAMILTONIAN PATH	1	2	2	1
11	3-COLORING, CHROMATIC NUMBER	1	2	2	1
12	CLIQUE COVER, PARTITION INTO TRIANGLES	1	1	1	1
13	HITTING SET	1	1	1	1
14	STEINER TREE	1	1	1	1
15	SIMPLE MAX CUT	1	1	1	1
16	DOMINATING SET	1	1	1	1
17	3SATWP, 3SATWN,	3	1	3	1

Table 1: Complexity of problems specified succinctly. The list is not exhaustive but rather serves as examples of the results that can be proved using the techniques developed here.

Acknowledgement: We thank Professors Ken Regan and Eric Allender, for constructive suggestions on an earlier draft of this paper. We also thank Professor Anne Condon, Ashish Naik, Ravi Sundaram and Sandeep Shukla for their help and suggestions during the course of writing this paper.

References

[AHU] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Reading MA., 1974.

[AC94] S. Agarwal and A. Condon, "On Approximation Algorithms for Hierarchical MAX-SAT," *Proc. of the 10th IEEE Conference on Structures in Complexity Theory,* June, 1995, pp. 214-226.

[BLT92] J.L. Balcazar, A. Lozano, and J. Toran " The Complexity of Algorithmic Problems for Succinct Instances," in *Computer Science* Ed. R. Baeza-Yates, 1992, pp. 351-377.

[BOW83] J.L. Bentley, T. Ottmann, P. Widmayer, "The Complexity of Manipulating Hierarchically Defined set of Intervals," *Advances in Computing Research, ed. F.P. Preparata* Vol. 1, (1983), pp. 127-158.

[CS81] P.R. Cappello and K. Steiglitz, "Digital Signal Processing Applications of Systolic Algorithms," *Proc. CMU Conference on VLSI Systems and Computations,* H.T Kung, B. Sproull and G Steele eds. 1981.

[CM93] E. Cohen and N. Megiddo, "Strongly Polynomial-time and NC Algorithms for Detecting Cycles in Dynamic Graphs," *Journal of the ACM (J. ACM)* Vol. 40, No. 4, September 1993, pp. 791-830.

[CM91] E. Cohen and N. Megiddo, "Recognizing Properties of Periodic graphs," *Applied Geometry and Discrete Mathematics,* Vol. 4, *The Victor Klee Festschrift,* P. Gritzmann and B. Strumfels, eds., ACM, New York, 1991, pp. 135-146.

[CLR] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, McGraw-Hill Book Co., 1990.

[FF58] L.R. Ford and D.R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," *Operations Research,* No. 6, 1958, pp. 419-433.

[Ga82] H. Galperin "Succinct Representation of Graphs," Ph.D. Thesis, Princeton University, 1982.

[GW83] H. Galperin and A. Wigderson, "Succinct Representation of Graphs," *Information and Control* , Vol. 56, 1983, pp. 183-198.

[Ga59] D. Gale, "Transient Flows in Networks," *Michigan Mathematical Journal*, No. 6, 1959 , pp. 59-63.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, San Francisco CA, 1979.

[GP+96a] J. Gu and P.W. Purdom and J. Franco and B.W. Wah, "Algorithms for Satisfiability (SAT) Problem: A Survey," *DIMACS Volume Series on Discrete Mathematics and Theoretical Computer Science: The Satisfiability (SAT) Problem*, American Mathematical Society, to appear.

[GP+96b] J. Gu and R. Puri and B. Du, "Satisfiability Problems in VLSI Engineering", *Discrete Applied Mathematics*, submitted, 1996.

[GP95] J. Gu and R. Puri, "Asynchronous Circuit Synthesis by Boolean Satisfiability," *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. 14, No. 8, Aug, 1995, pp. 961-973.

[Gu94] J. Gu, "Optimization Algorithms for the Satisfiability (SAT) Problem, " *Advances in Optimization and Approximation.* Ding-Zhu Du and Jie Sun (ed), Kluwer Academic Publishers, Boston, MA, 1994, pp. 72-154.

[Gu92] J. Gu, "The *UniSAT* Problem Models (Appendix)," *IEEE Trans. on Pattern Analysis and Machine Intelligence*," Vol. 14, No. 8, Aug, 1992, pp. 865.

[HK87] A. Habel and H.J. Kreowski, " May we Introduce to you: Hypergraph Languages Generated by Hyperedge Replacement," *Proc. 13th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'87)* , Springer Verlag, LNCS, Vol. 291, 1987, pp. 15-26.

[HLW92] F. Höfting, T. Lengauer and E. Wanke, "Processing of Hierarchically Defined Graphs and Graph Families," in *Data Structures and Efficient Algorithms* (Final Report on the DFG Special Joint Initiative), Springer-Verlag, LNCS 594, 1992, pp. 44-69.

[HW95] F. Höfting and E. Wanke, "Minimum Cost Paths in Periodic Graphs," *SIAM Journal on Computing*, Vol. 24, No. 5, October 1995, pp. 1051-1067.

[HW94] F. Höfting and E. Wanke, "Polynomial Time Analysis of Toroidal Periodic Graphs," *Proc. of International Colloquium on Automata, Programming and Languages,* LNCS, July, 1994.

[Hu73a] H.B. Hunt III, "On The Time Complexity of Languages," *Proc. 5th Annual ACM Symposium on Theory Of Computing, (STOC)*, 1973, pp. 10-19.

[HMS94] H. B. Hunt III, M. V. Marathe and R. E. Stearns, "Generalized CNF Satisfiability Problems and Non-Efficient Approximability," *Proc. 9th ACM Conf. on Structure in Complexity Theory*, June-July 1994, pp. 356-366.

[IS86] K. Iwano and K. Steiglitz, "Optimization of one-bit full adders embedded in regular structures," *IEEE Transactions on Acoustics, Speech and Signal Processing,* 1986.

[IS87] K. Iwano and K. Steiglitz, "Testing for Cycles in Infinite Graphs with Periodic Structure," *Proc. 19th Annual ACM Symposium on Theory of Computing, (STOC)*, 1987, pp. 46-53.

[IS88] K. Iwano and K. Steiglitz, "Planarity Testing of Doubly Connected Periodic Infinite Graphs," *Networks*, No. 18, 1988, pp. 205-222.

[JL77] N.D.Jones and W.T. Laaser, "Complete Problems for Deterministic Polynomial Time," *Theoretical Computer Science*, No. 3, 1977, pp. 105-117.

[KMW67] R.M. Karp, R.E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," *Journal of the ACM (J. ACM)*, Vol. 14, No. 3, 1967, pp. 563-590.

[Ka72] R.M. Karp, "Reducibility Among Combinatorial Problems," in R.E. Miller and J.W. Thatcher (eds.) *Complexity of Computer Computations*, Plenum Press, N.Y. 1972, pp. 85-103.

[KO91] M. Kodialam and J.B. Orlin, "Recognizing Strong Connectivity in Periodic graphs and its relation to integer programming," *Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1991, pp. 131-135.

[KS88] K. R. Kosaraju and G.F. Sullivan, "Detecting Cycles in Dynamic Graphs in Polynomial Time," *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1988, pp. 398-406.

[Le82] T. Lengauer, "The Complexity of Compacting Hierarchically Specified Layouts of Integrated Circuits," *Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, 1982, pp. 358-368.

[Le86] T. Lengauer, "Exploiting Hierarchy in VLSI Design," *Proc. AWOC '86*, Springer-Verlag, LNCS 227, 1986, pp. 180-193.

[LW87a] T. Lengauer and E. Wanke, "Efficient Solutions for Connectivity Problems for Hierarchically Defined Graphs ," *SIAM Journal on Computing*, Vol. 17, No. 6, 1988, pp. 1063-1080.

[Le88] T. Lengauer, "Efficient Algorithms for Finding Minimum Spanning Forests of Hierarchically Defined graphs," *Journal of Algorithms*, Vol. 8, 1987, pp. 260-284.

[Le89] T. Lengauer, "Hierarchical Planarity Testing," *Journal of the ACM (J. ACM)* , Vol. 36, No. 3, July 1989, pp. 474-509.

[LW92] T. Lengauer and K.W. Wagner, "The Correlation Between the Complexities of Non-Hierarchical and Hierarchical Versions of Graph Problems," *Journal of Computer and System Sciences (JCSS)*, Vol. 44, 1992, pp. 63-93.

[LW93] T. Lengauer and E. Wanke, "Efficient Decision Procedures for Graph Properties on Context-Free Graph Languages," *Journal of the ACM (J. ACM),* Vol. 40, No. 2, 1993, pp. 368-393.

[Li82] D. Lichtenstein, "Planar Formulae and their Uses", *SIAM Journal on Computing*, Vol 11, No. 2, May 1982 , pp. 329-343.

[MH+93a] M.V. Marathe, H.B. Hunt III, and S.S. Ravi, "The Complexity of Approximating PSPACE-Complete Problems for Hierarchical Specifications," *Nordic Journal of Computing*, Vol. 1, 1994, pp. 275-316.

[MR+93] M.V. Marathe, V. Radhakrishnan, H.B. Hunt III, and S.S. Ravi, "Hierarchical Specified Unit Disk Graphs," *Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '93)*, June, 1993, pp. 21-32. to appear in *Theoretical Computer Science*, March 1996.

[MH+94] M.V. Marathe, H.B. Hunt III, R.E. Stearns and V. Radhakrishnan, "Approximation schemes for PSPACE-Complete problems for succinct graphs," *Proceedings of 26th Annual ACM Symposium on the Theory of Computing (STOC),* May 1994, pp. 468-477.

[Ma94] M.V. Marathe *Complexity and Approximability of NP- and PSPACE-hard Optimization Problems,* Ph.D. thesis, Department of Computer Science, University at Albany, Albany, NY August, 1994.

[MH+95a] M.V. Marathe, H.B. Hunt III, R.E. Stearns and D. J. Rosenkrantz, "Approximation Algorithms for NEXPTIME-Complete Periodically Specified Problems," submitted, 1995.

[MH+95b] M.V. Marathe, H.B. Hunt III, R.E. Stearns and D. J. Rosenkrantz, "Periodically Specified Satisfiability Problems: An Alternative to Domino Problems," manuscript Nov 1995.

[MH+95c] M.V. Marathe, H.B. Hunt III and R.E. Stearns "A Uniform Approach to prove hardness for problems specified by G.C.R, S.C.R and BOW Specifications," manuscript Nov. 1995.

[MH+96] M.V. Marathe, H.B. Hunt III, R.E. Stearns and V. Radhakrishnan, "Complexity of hierarchically and 1-dimensional periodically specified Problems II: Easiness Results and Dichotomy Theorem," in preparation, March 1996.

[MTM92] J.O. McClain, L.J. Thomas and J.B. Mazzola, *Operations Management,* Prentice Hall, Englewood Cliffs, 1992.

[MC80] C. Mead and L. Conway, *Introduction to VLSI systems* Addison Wesley, 1980.

[MS81] B.Monien and I.H.Sudborough, "Bounding the Bandwidth of NP-Complete Problems," *Proc. 13th ACM Annual Symposium on Theory of Computing (STOC)*, 1981, pp. 279-292.

[Or82a] J.B. Orlin, "The Complexity of Dynamic/Periodic Languages and Optimization Problems," Sloan W.P. No. 1679-86 July 1985, Working paper, Alfred P. Sloan School of Management, MIT, Cambridge, MA 02139. A Preliminary version of the paper appears in *Proc. 13th ACM Annual Symposium on Theory of Computing (STOC)*, 1978, pp. 218-227.

[Or84b] J.B. Orlin, "Some Problems on Dynamic/Periodic Graphs," *Progress in Combinatorial Optimization*, Academic Press, May 1984, pp. 273-293.

[PY86] C. Papadimitriou and M. Yannakakis, "A note on Succinct Representation of Graphs," *Information and Computation* No. 71, 1986, pp. 181-185.

[Pa94] C. Papadimitriou, *Computational Complexity* Addison-Wesley, Reading, Massachusetts, 1994.

[RH93] D.J. Rosenkrantz and H.B. Hunt III, "The Complexity of Processing Hierarchical Specifications", *SIAM Journal on Computing*, Vol. 22, No. 3, 1993, pp. 627-649.

[Sa70] W.J.Savitch, "Relationships Between Nondeterministic and Deterministic Tape Complexities," *Journal of Computer and System Sciences (JCSS),* Vol. 4, No. 2, pp. 177-192, 1970.

[Sc78] T. Schaefer, "The Complexity of Satisfiability Problems," *Proc. 10th ACM Symposium on Theory of Computing (STOC)*, 1978, pp. 216-226.

[Sc78a] T. Schaefer, "On the Complexity of Some Two-Person Perfect Information Games," *Journal of Computer and System Sciences (JCSS),* 1978, pp. 185-225.

[Sc76] C. Schnorr "Satisfiability is quasi-linear complete for NQL," *Journal of the ACM (J. ACM),* No. 25, 1976, pp. 136-145.

[SH90] R. E. Stearns and H. B. Hunt III "Power Indices and Easier Hard Problems," *Mathematical Systems Theory* Vol. 23, 1990, pp. 209-225.

[Wa86] K.W. Wagner, "The Complexity of Combinatorial Problems with Succinct Input Representation," *Acta Informatica* , Vol. 23, No. 3, 1986, pp. 325-356.

[Wa93] E. Wanke, "Paths and Cycles in Finite Periodic Graphs," *Proc. 20th Symposium on Math. Foundations of Computer Science (MFCS)*, LNCS 711, Springer-Verlag, 1993, pp. 751-760.

[Wi90] M. Williams, "Efficient Processing of Hierarchical Graphs ," *TR 90-06*, Dept of Computer Science, Iowa Sate University. (Parts of the report appeared in WADS'89, pp. 563-576 and SWAT'90, pp. 320-331 coauthored with Fernandez-Baca.)

Los Alamos National Laboratory P.O. Box 1663, MS B265, Los Alamos NM 87545

E-mail address: `madhav@c3.lanl.gov`

Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222, USA.

E-mail address: `hunt@cs.albany.edu`

Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222, USA.

E-mail address: `res@cs.albany.edu`

Part of the work was done while the author was at SUNY Albany. Current Address: Mailstop 47LA, Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, California 95014-9913.

E-mail address: `rven@cup.hp.com`